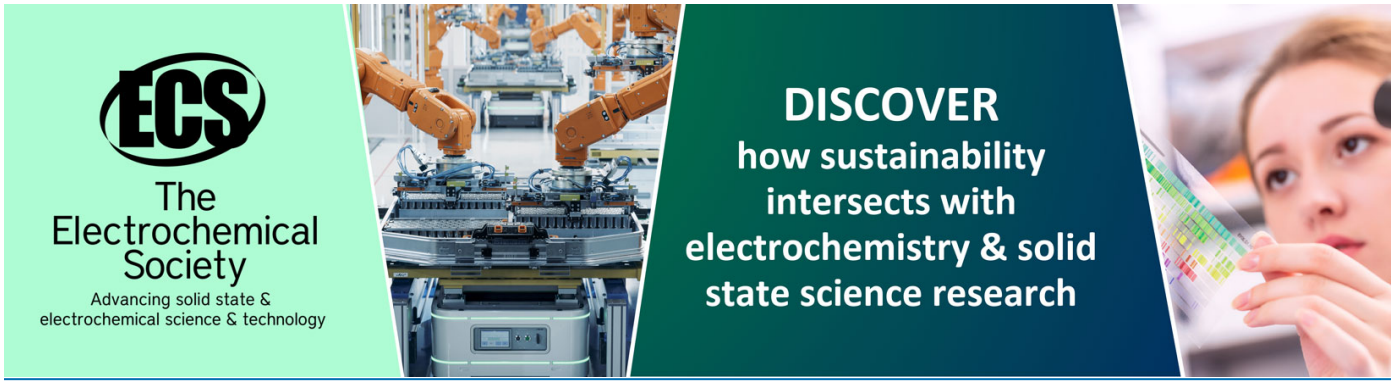
Design of Operating Platform for Intelligent Charging Pile Based on Micro-service



Design of Operating Platform for Intelligent Charging Pile Based on Micro-service

Xueyuan Pan and Shejiao Hu

Information and Communication Engineering, Hefei University of Technology, School of Computer and Information, roomA1001.
Email: 1761309363@qq.com

Abstract. The introduction of "new energy vehicle charging pile" as one of the contents of "new infrastructure" indicates that the field of charging pile is facing a new round of technological innovation, deployment difficulties and other shortcomings are no longer suitable for its charging operation of the growing business needs. In view of this, a design method based on microservice architecture is proposed. The open source distributed framework Spring Cloud and its components are used to realize the mechanism of service registration discovery, load balancing, service fault tolerance and so on. The reformed operation system has a clear structure, stable operation, timely monitoring and processing of abnormal information such as charging pile fault, and has a broad application prospect.

1. Introduction

The 2020 meeting of the Politburo made it clear that new forms of infrastructure, including new energy vehicle charging piles, would need to be stepped up to provide new impetus to economic growth. "New infrastructure" drive the field of charging piles in all aspects, not only to promote the construction of high-quality field stations and public piles, but also to operators and charging operation platform put forward more intelligent, humane requirements.

Most of the traditional charging platforms adopt single architecture or service-oriented architecture, which centralizes all functional modules in the same application and adopts unified programming language. With the continuous expansion of the business scope of charging pile, the overall complexity of the system continues to increase, the shortcomings of the traditional architecture gradually exposed. Poor scalability, small changes to a function that affect the deployment of the entire application, database data volume and confusion, low resource utilization, and other issues not only affect the user experience, it is not conducive to data monitoring, fault analysis and control of the whole charging process. Therefore, the construction of a new charging service platform is an urgent problem to be solved at this stage.

This paper presents a new charging pile operation service platform based on micro-service architecture and a continuous deployment of software system based on Docker container technology, which has the advantages of high availability, high scalability and rapid replacement. Following the trend of new software engineering concepts such as agile development, DevOps, continuous delivery, etc. , charging studs are becoming "smart, professional, and contextual" as cloud computing and Intel Virtualization Technology mature.

2. Microservice Architecture and Docker

2.1. Microservice

Fowler and other computer scientists came up with the concept of microservice. The core idea of microservice is to segment services around the Organization of business functions, and each service can be developed and implemented in the appropriate technology and language according to its own functional requirements, independent operation and process; microservices tend to reinforce terminals and weak channels, and are committed to forming highly cohesive and low-coupling service units, with distinct boundaries between services and a REST-style lightweight messaging mechanism, cooperate with each other to complete the function of the whole system. The advantages of micro-service architecture are: low development cost, good fault tolerance, simple and quick deployment.

The current mainstream distributed service frameworks include Spring Cloud and Dubbo. In contrast to Dubbo, the Spring Cloud framework and its core components cover every aspect of the functionality of microservices, including service governance, and the community is active. Therefore, Spring Cloud is chosen as the micro-service technology to realize the design of the whole system function.

2.2. Spring Cloud

Spring Cloud is an ordered set of Spring Boot-based frameworks and components. It provides many tools for developers to quickly build a common pattern of distributed systems, such as service registration discovery, load balancing, link tracing and so on. The complete Spring Cloud framework is shown in Figure 1 below:

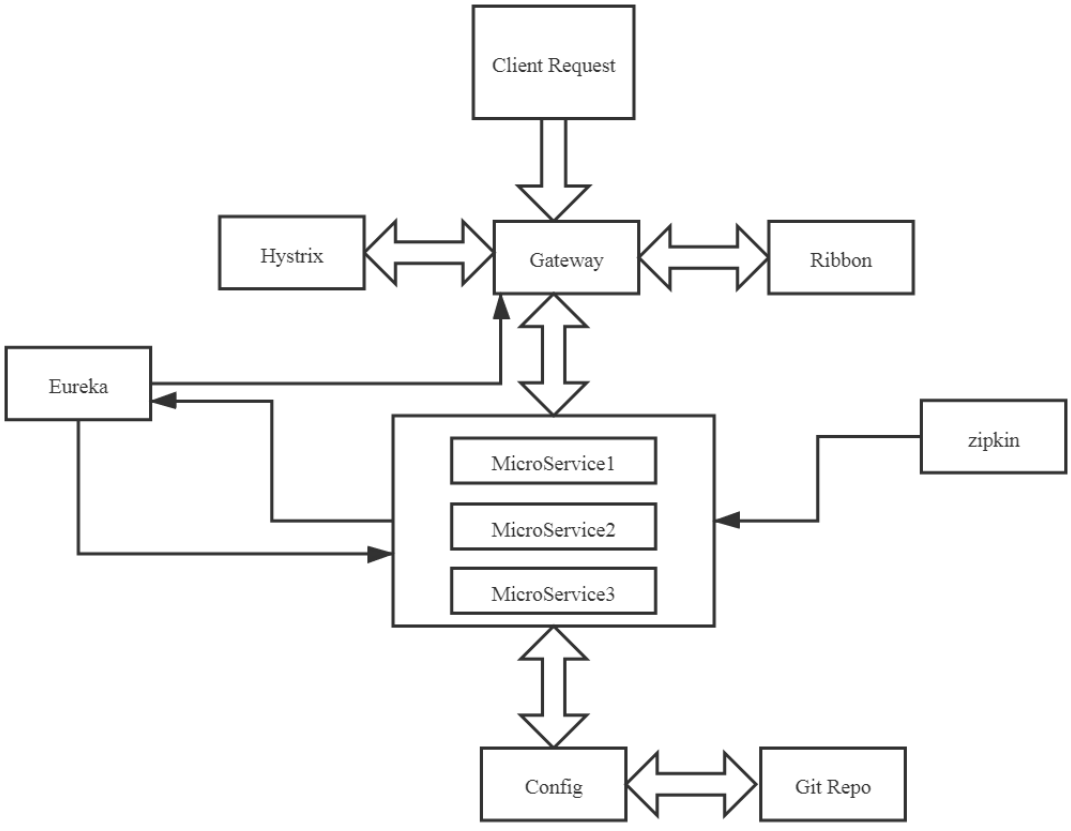


Figure 1. Spring Cloud overall framework

(1) Spring Cloud Eureka implements service governance. The core of service governance consists of service provider, service consumer and registry. The Eureka Server component defines the service registry, and the microservice to be registered connects to Eureka Server via the Eureka Client component to complete the registration.

(2) Spring Cloud Ribbon for load balancing. Spring Cloud Ribbon, based on Netflix Ribbon, mainly provides client software load balancing Algorithm and service call function to achieve high system availability.

(3) Hystrix implements fault-tolerant mechanism and provides service degradation, service fusing and near real-time monitoring functions to ensure pre-processing of error cases to avoid cascading failures and improve the flexibility of distributed systems.

(4) Spring Cloud Config is a distributed configuration center that provides configuration services through the server side for multiple configuration centers, dynamically updates configuration during runtime, and uses Git by default to store configuration files.

(5) the Spring Cloud Gateway Component implements the routing Gateway. Shield server implementation logic, provide a unified user interface. Its internal integration of the fuse, authentication, flow control and other functions.

(6) Spring Cloud Zipkin for distributed link tracking. By collecting and tracking the request data in the system, the operator can monitor the time spent by the request in each micro-service more quickly.

2.3. Docker

Because the virtual machine has many shortcomings such as resource occupation, redundant steps, and time-consuming, Linux virtualization technology came into being. Docker is a virtualization technology that solves operating environment and configuration problems and helps continuous integration and overall release. Compared with traditional virtual machines, it has the following advantages:

(1) Efficient and lightweight. A container is different from a virtual machine running a complete set of operating systems, and only requires the library and configuration environment required for software to run.

(2) Rapid deployment and delivery. Write the dockfile file to build the Docker image and deliver it to the warehouse, pull the image in another environment and instantiate it. Build once, run anywhere.

(3) Simplify system operation and maintenance. The development-test-production environment is integrated, and the inconsistency of the underlying infrastructure and operating system will not cause any impact, which greatly reduces the workload of operation and maintenance personnel.

(4) Improve the efficiency of computing resource utilization. Docker is kernel-level virtualization. It can run hundreds of containers on a physical machine, greatly improving the CPU and memory utilization of physical servers.

In the actual deployment process, the configuration environment required by the sub-application is converted into instructions and written into the dockerfile file to run the file to generate the image. Write Docker Compose.yml file to arrange service scheduling and run mirroring.Based on this, the combination of microservices and Docker technology is used to achieve rapid deployment and continuous integration.

3. System Requirement Analysis and Overall Structure

The intelligent charging pile operation platform is based on the design and implementation of the microservice architecture proposed in this paper.Adopt front-end and back-end separation mode, and use json for data interaction. The client includes a mobile WeChat applet and a browser. It adopts a restful-style HTTP request-response mechanism, and comprehensively considers user experience, system performance, operation management and other factors to achieve a more intelligent, personalized, and operating platform for the charging pile system . Figure 2 is a diagram of the overall system architecture.The smart charging pile operating platform shall have the following functions:

For ordinary users of charging pile:

(1) users log in with small programs on the mobile side, and the system provides them with cloud card query, charging station search and navigation, start-stop and reservation charging pile, wechat charging payment and other functions through a unified authentication platform.

(2) the charging pile is pushed to the client in time when the charging process is abnormal

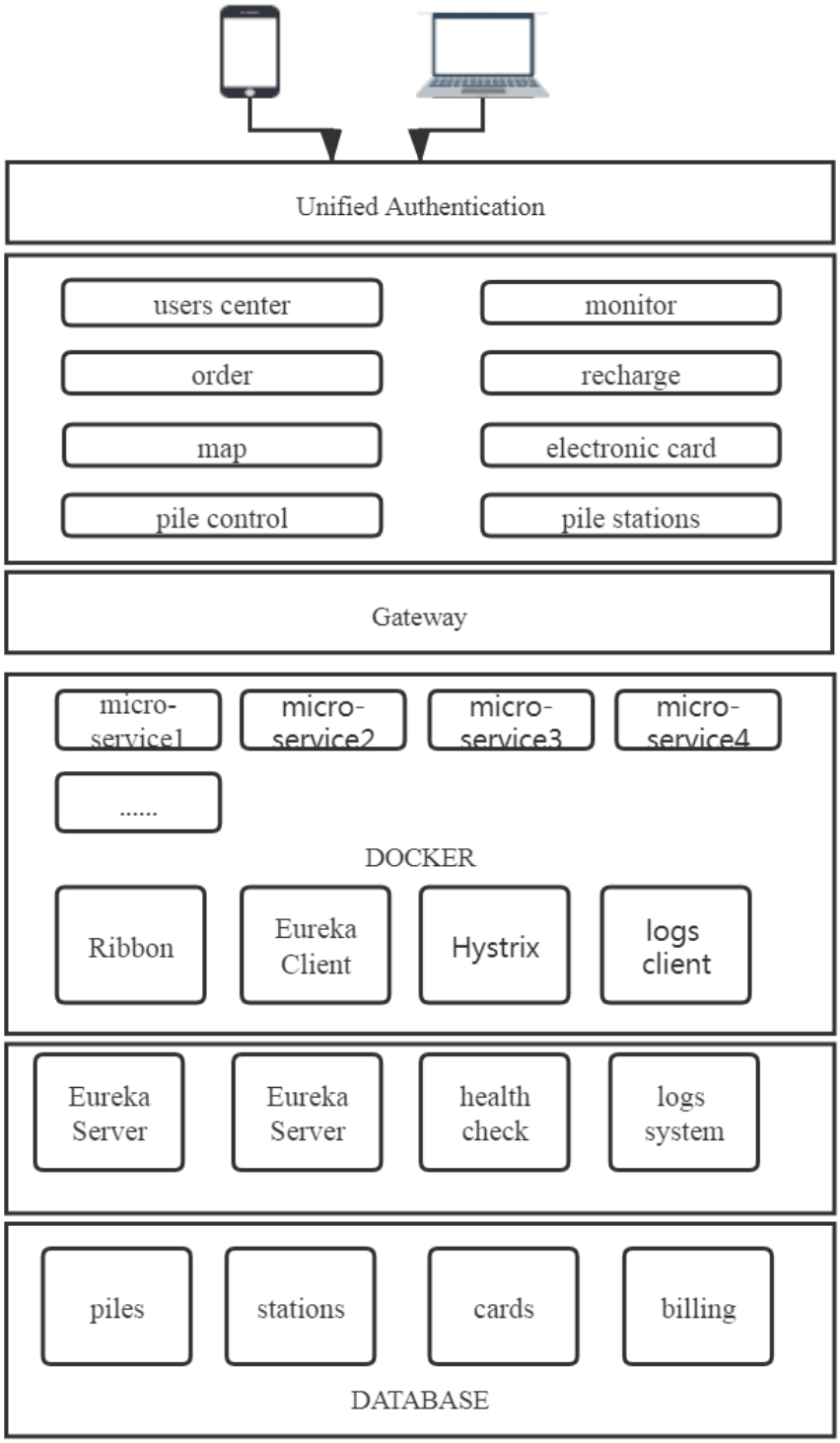


Figure 2. System architecture

For charging pile operators, the system should perform the following functions:

(1) charging pile management: After charging pile is connected to the system platform, the equipment is authorized to collect, store, analyze and monitor its status data. The operator can remotely control the start and stop of the charging posts. To provide four ways to start: automatic filling, according to the time start, according to the power start, according to the amount of start.

(2) cloud card management: One user for one card. The operating platform provides a free e-card for the first time users who log in to a small program. Users can charge, deduct and cancel the card. When the card balance is less than a certain value, the operating platform should be able to remind users to recharge before the next use.

(3) charging station management: The map on the first page of the operation platform distributes the charging stations in various districts and the usage of charging piles in the stations, including the number of idle/working posts, the total number of alarm calls for charging posts, the number of orders for completed transactions and the amount of money.

(4) Order System: The operator may inquire the historical order and the pending order detailed information. According to the order number, charging post number, charging gun muzzle number and so on for accurate inquiry.

(5) billing system: The operating platform sets the charging rates according to the pricing strategy for Peak Pinggu District Electricity consumption at different times of the day and the charging charges required for charging posts in different districts.

(6) monitoring system: Two dimensions, business logic and physical machine resource parameters, comprehensively monitor the service operation of the system platform. Real-time monitoring of the charging post in operation, including the current SOC, the value of charging voltage and current, BMS information and so on. Once the value exceeds the safe range, stop charging the stake immediately to warn. Warning information will be sent to the field personnel by SMS, email and other means and Wechat to notify users of the end of the order. Through the link tracking to accurately locate the Error Service, through the query log to get the details of the error information and timely inform the field technical personnel for processing.

4. Concrete Realization

4.1. Service Registration and Discovery

Spring Cloud realizes service registration and discovery through Eureka Server and Eureka Client, where Eureka Client separates service provider and service consumer. The Eureka Server as registry is implemented in three steps: Create a Module to place on pom. XML, add a configuration file to your application. XML, create the startup class and note the@EnableEurekaServer annotation. The Eureka Client implementation differs from the Eureka Server implementation in that it adds the@EnableEurekaClient annotation to the service consumer when creating startup classes.

Using client discovery mechanism, service consumers directly invoke service providers, simple architecture, no additional costs. At the same time, in order to ensure the high availability of Eureka, Eureka registry cluster is constructed, and the service list data of each registry is shared synchronously to realize the load balance and fault tolerance of service governance.

4.2. Spring Cloud Gateway

The Spring Cloud Gateway infrastructure uses a high-performance reactor-mode communication framework, and therefore replaces Zuul in this design. It provides routing and forwarding and implements the role of the filter chain gateway. The overall workflow is for the client to send a request to the Spring Cloud Gateway, which is routed to the handler mapping in the Gateway (the match condition is called predicate) , and then to the web handler if the match is successful, the handler finishes by sending the request through a series of filter chains to the actual business execution unit and returns the result.Spring Cloud Gateway configuration in the YML file, add the@EnableEurekaClient annotation to the primary startup class.

4.3. Load Balancing

Spring Cloud Ribbon is in-process client soft-load balancing for controlling HTTP requests, which is used in conjunction with the RSET TEMPLATE to implement the rest access service. According to the list of callable services provided by Eureka Server, Ribbon selects the calling instance to realize the local RPC remote service call based on the load balancing algorithm. Taking into account the actual use of charging pile operation platform, this paper adopts the available filterrule load balancing Algorithm, the core idea is to filter out the failure cases first, and then select the cases with a small number of concurrency.

4.4. Service Reliability Assurance

The Spring Cloud Hystrix component exists to prevent service avalanches in distributed systems. Its core functions are as follows:

(1) service degradation. When the downstream business stops working, the upstream business should be demoted to keep the core business running normally. Configuration via@HystrixCommand.

(2) service fuse. When a service fails to be accessed multiple times, the fuse is turned on and a health check is sent to see if the connection can be restored.

(3) service restriction. The self-protection mechanism of the service adopts many flow-limiting strategies, such as discarding the redundant requests or rejecting the service which produces a large number of requests, in order to prevent the service from overloading in a short time.

(4) visual surveillance. Visual data monitoring is declared using the@Hystrix DashBoard with@Spring Boot Actuator and via Hystrix. The stream node gets the details of the monitoring requests and presents them to the operators in real time in the form of statistical charts.

The internal invocation relationship in microservice is complicated, the Spring Cloud Zipkin component is used to locate and trace the problem, the@EnablezipkinServer declaration is used to start the Zipkinserver, and the log analysis component e (ElasticSearch) l (Logstash) k (Kibana) is used to search for the problem and show it to the operators.Using redis cache, the intelligent charging post platform designed in this paper can monitor memory value, data flow, MySQL database and disk space and connection number. The monitoring strategy for the data collected during the charging process of the charging post is to deploy an index collector to query the important data indexes in real time, draw the monitoring interface and give an alarm when the data exceeds the threshold value.

Based on the above monitoring measures, ensure the smooth and safe operation of charging pile service.

5. Analysis of Test Results

5.1. Experimental Environment

The software environment installed in this lab is shown in Table 1.

Table 1. Experiment software environment

name	version
Docker	19.03.05
OS	Centos7.0
Spring Cloud	Alibaba 2.1.0
Java	27.9
Spring Boot	2.2.2
Maven	3.5

5.2. Performance Comparison

The stress testing tool Apache Jmeter is used to evaluate the performance of this system and the traditional single application system. Under the premise of high concurrency, the same URL is requested,set the number of concurrent to 1000. The comparison of network indicators is shown in the

following Table 2. From the data in Table 2, it can be concluded that the performance of the smart charging pile operating system based on microservice is greatly improved.

Table 2. A Comparison of network indicators under the two architectures

Index	Monolithic architecture	Microservice architecture
concurrency	1000	1000
Time for per request(ms)	1.6754	0.2364
Request nums/s	734.54	2999.48
Transfer rate	221.47received	593.77received

6. Concluding Remarks

This paper designs a new smart charging pile operating platform based on the microservice architecture. Compared with the traditional single application architecture, its system has high availability and high scalability. The combination of SpringCloud and Docker container technology solves the complex deployment problems, which is beneficial to System maintenance and replacement. Therefore, in today's rapid development of the field of new energy charging piles, this system has good promotion value and significance.

7. References

[1] Long Xinzheng,Peng Yiming and Li Ruomiao 2017.Information service platform based on microservices framework.*Journal of SouthEast University* J.**47** 48-52
[2] Li Qing and Hu Shejiao 2019.Electric Vehicle Intelligent Charging System Based on NB-IOT.*Measurement and Control Technology J.***38** *53-57*
[3] Wu Xia and AI Fangju 2020.Crowdfunding system based on microservices architecture.*Journal of Hubei University* J.**42** 103-8
[4] Ouyang Rongbin,Wang Qianyi and Long Xinzheng 2016.A data service framework based on micro-services.*Journal of Huazhong University of Science and Technology* J.**44** 126-130
[5] Zhang Jing,Huang Xiaofeng and Li ChunYang 2017.Design and Implementation of Microservice Architecture. *Csa* J .**26** 259-262
[6] Zhao Yu 2020. Multi-Language Micro-Service Platform Based on ServiceComb.*Csa* J **29** 84-91
[7] Xiao Yao and Zhu Zhixiang 2019.Design and Implementation of a Monitoring System Based on Docker.*Computer&Degital Engineering* J. **47** 2919-25
[8] Liu Shuyang,Zhang Manyi and Cao Qiang 2018.Design and Implementation of a Docker dynamic scheduling algorithm.*Computer Engineering&Science* J. **40** 2112-19
[9] Wu Huayao and Deng Wenjun 2020.Research Progess on the Development of Microservices.*Journal of Computer Research and Development* J. **57** 525-541
[10] Li Chunyang,Liu Di and Cui Wei 2017.Unified application development platform based on micro-service architecture .*Computer System &Applications* J .**26** 43-48
[11] Huang Qiqi,Xiang Qian,Cheng Maoshang and Lv Zhijun 2020.Warehouse Management and Control System Based on Microservices. *Journal of Donghua University* J .**46** 83-90